Digital Electronics

Chapter 3(part - 2)
Combinational Logic

Outline of Chapter 3(2)

- 5.1 Binary Parallel Adder
- 5.2 Decimal Adder
- 5.3 Magnitude Comparator
- 5.4 Decoder
- 5.5 Multiplexer

BINARY PARALLEL ADDER

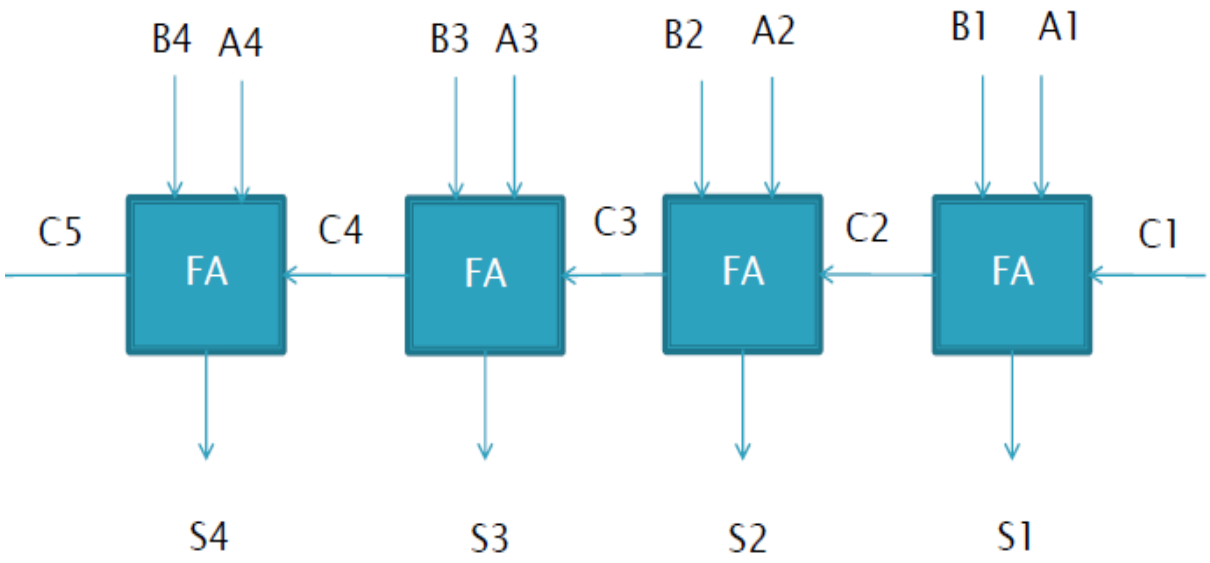
- The full-adder forms the sum of two bits and a previous carry.
- Two binary numbers of n bits each can be added by means of this circuit.
- When pair of bits are added through the full adder, the circuit produces a carry to be used with the pair of bits one higher significant position.\
- The bits are added with full-adders, starting from the least significant position, to form the sum bit and carry bit.
- The sum of two n-bit binary numbers A and B can be generated in two ways: either in serial fashion or in parallel.

BINARY PARALLEL ADDER

- The sum of two n-bit binary numbers A and B can be generated in two ways: either in serial fashion or in parallel.
- The serial addition method uses only one full adder circuit and a storage device to hold the generated output carry and sum.
- The parallel method uses n full-adder circuit.
- A binary parallel adder is a digital function that produces the arithmetic sum of two binary numbers in parallel.

BINARY PARALLEL ADDER

- Example:



BINARY PARALLEL ADDER

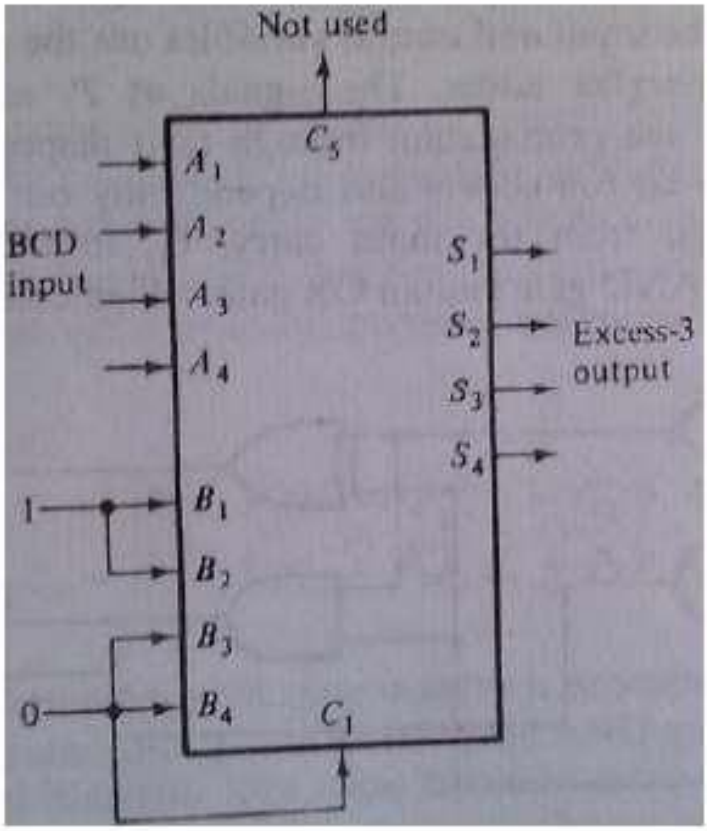
- An n-bit parallel adder requires n full-adders.
- It can be constructed from 4-bit, 2-bit and 1-bit full-adders ICs by cascading several packages.
- The 4-bit binary parallel adder is a typical example of an MSI function.
- It can be used in many applications involving arithmetic operations.

BINARY PARALLEL ADDER

- The application of this MSI function to the design of a combinational circuit is demonstrated in the example of BCD to excess-3 code converter.

- Example:

BCD to Excess-3 Code converter

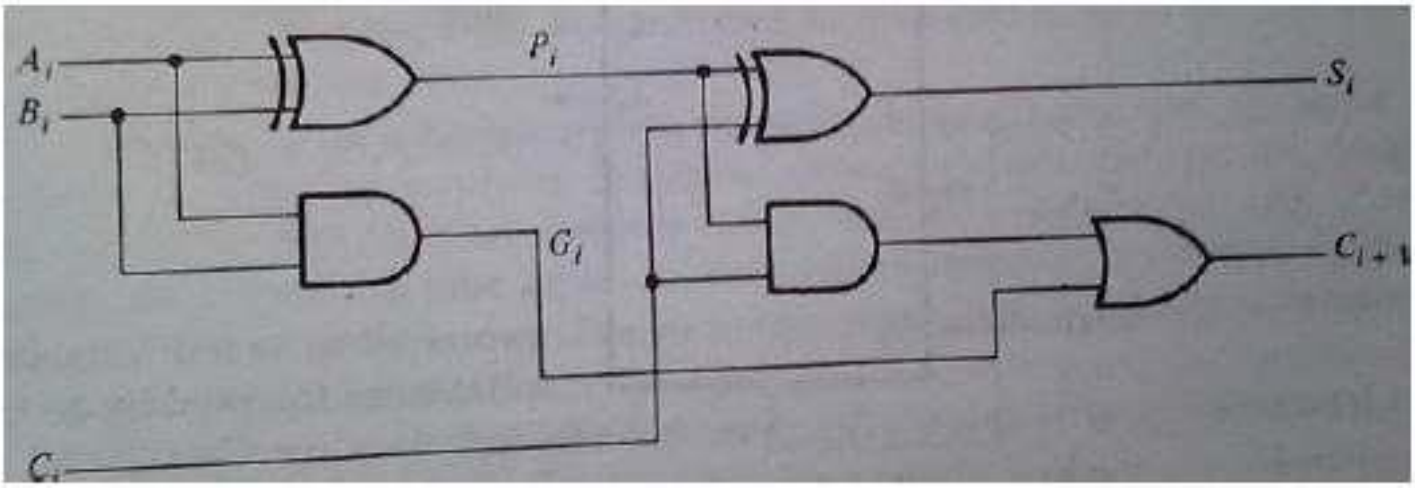


CARRY PROPOGATION

- The addition of two binary numbers in parallel implies that all the bits of the augend and the addend are available for computation at the same time.
- As in any combinational circuit, the signal must propagate through gates before the correct output sum is available in output terminals.
- The total propagation time is equal to the propagation delay of typical gate times the number of gate levels in the circuit.
- The longest propagation delay time in a parallel adder is the time it takes the carry to propagate through the full-adders.

CARRY PROPOGATION

- The number of gate levels for the carry propagation can be found from the circuit of the full adder.
- The signal from the carry (C_i) to the output carry (Ci+1) propagates through 2 gate levels.



CARRY PROPOGATION

- If there are four full-adders in the parallel adder, the output carry C5 would have 2*4=8 gate levels from C1 to C5.
- The total propagation time in the adder would be the propagation time in one half adder plus eight gate levels.
- For an n-bit parallel adder, there are 2n gate levels for the carry to propagate through.
- The carry propagation time is a limiting factor on the speed with which two numbers are added in parallel.

CARRY PROPOGATION

- All other arithmetic operations are implemented by successive additions, the time consumed during the addition process is very critical.
- One way to reduce the carry propagation delay time is to employ faster gates with reduced delays.
- Another solution is to increase the equipment complexity in such a way that the carry delay time is reduced.
- The most widely used technique employs the principle of look-ahead carry.

CARRY PROPOGATION

- Look-ahead carry:
- If we define two variables:

$$P_i = A_i \oplus B_i$$
$$G_i = A_i B_i$$

- Gi is called a carry generated and it produced an output carry when both Ai and Bi one.
- Pi is called a carry propagate because it is the term associated with the propagation of the carry C_i to C_i+1
- The output sum and carry can be expressed as

$$S_i = P_i \oplus C_i$$
$$C_{i+1} = G_i + P_i C_i$$

CARRY PROPOGATION

- The Boolean functions for the carry output of each stage are:

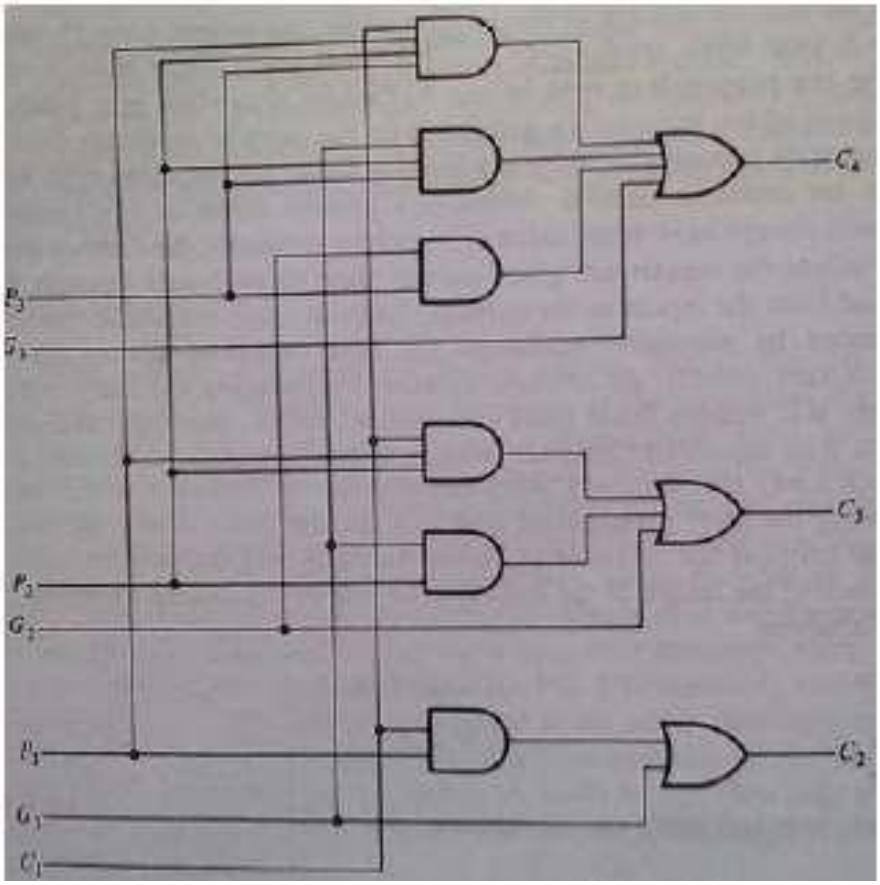
$$C_2 = G_1 + P_1C_1$$

$$C_3 = G_2 + P_2C_2 = G_2 + P_2(G_1 + P_1C_1) = G_2 + P_2G_1 + P_2P_1C_1$$

$$C_4 = G_3 + P_3C_3 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1C_1$$

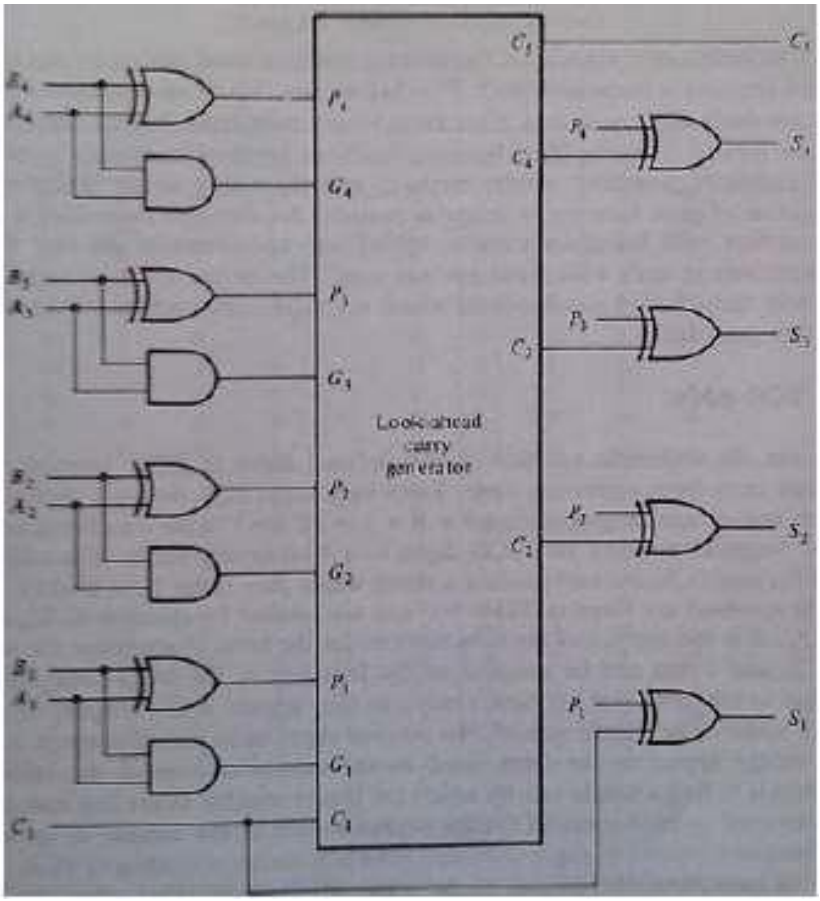
CARRY PROPOGATION

- Circuit diagram of a look–ahead carry generator:



CARRY PROPOGATION

- 4-bit Full-adders with look-ahead carry



DECIMAL(BCD) ADDER

- A decimal adder requires a minimum of nine inputs and five outputs, since four bits are required to code each decimal digit and the circuit must have an input and output carry.

- Consider the arithmetic addition of two decimal digits in standard BCD code (8421 code), together with an input carry from a previous stage.

- Since each input digit does not exceed 9, the output sum cannot be greater than 9+9+1 =19, the 1 in the sum being an input carry.

- Apply 2 BCD digits to a 4-bit binary adder.

DECIMAL(BCD) ADDER

- The adder will form the sum in binary and produce a result that ranges from 0 through 19.

- These binary numbers are listed in the table and are labelled by K, Z8,Z4,Z2,Z1 and K is carry.

- The columns under the binary sum list the binary value that appears in the outputs of the 4-bit binary adder

DECIMAL(BCD) ADDER

Derivation of a BCD Adder

Binary Sum					BCD Sum					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

DECIMAL(BCD) ADDER

- From the table, when the binary sum is equal to or less than 1001, the corresponding BCD number is identical, and therefore no conversion is needed.

- When the binary sum is greater than 1001, non valid BCD representation is obtained.

- Addition of binary 6 (0110) to the binary sum converts it to the correct BCD representation and also produces an output carry as required.

DECIMAL(BCD) ADDER

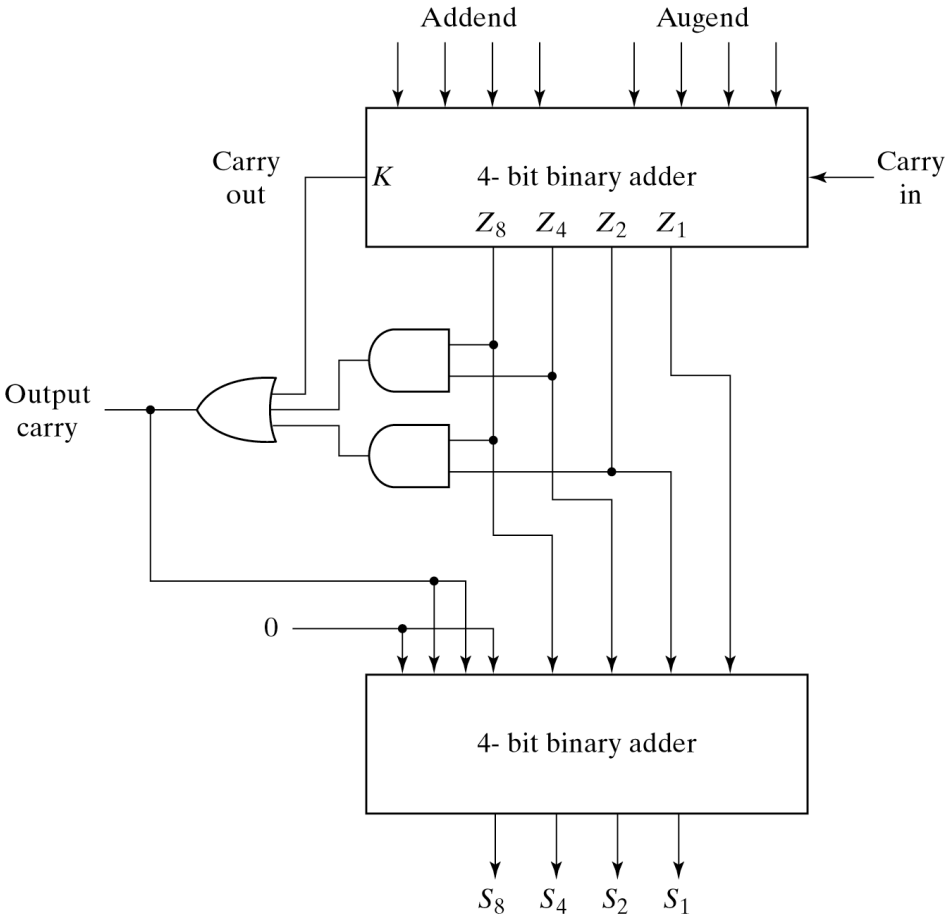
- A correction is needed when the binary sum has an output carry K=1. The other six combinations from 1010 through 1111 that need a correction have a 1 in position Z8. To distinguish them from binary 1000 and 1001, which also have a 1 in position Z8, it can be concluded that Z 4 or Z2 must have a 1.

- Therefore, the condition for a correction and an output carry can be expressed by the Boolean function:

  C=K+Z8 Z 4+Z8 Z2

- When C =1, it is necessary to add 0110 to the binary sum and provide an output carry for the next stage.

DECIMAL(BCD) ADDER



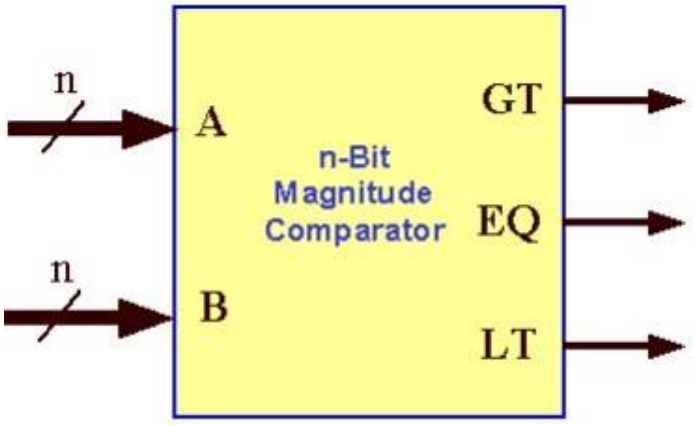
MAGNITUDE COMPERATOR

- A magnitude digital comparator is a combinational circuit that compares two digital or binary numbers (consider A and B) and determines their relative magnitudes in order to find out whether one number is equal, less than or greater than the other digital number.

- Three binary variables are used to indicate the outcome of the comparison as A>B, A<B, or A=B.

MAGNITUDE COMPERATOR

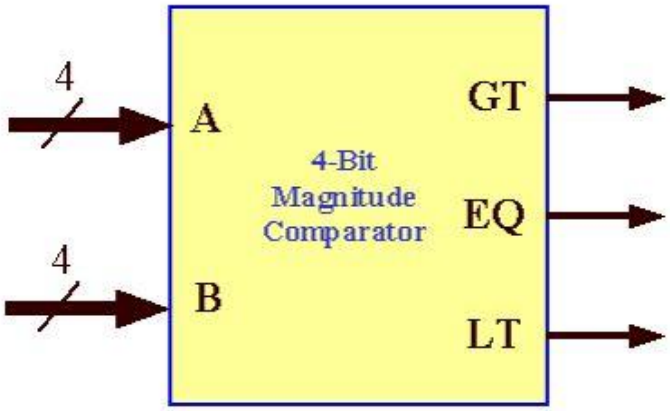
- The below figure shows the block diagram of a n-bit comparator which compares the two numbers of n-bit length and generates their relation between themselves.



MAGNITUDE COMPERATOR

4-bit Magnitude Comparator :

- Inputs: 8-bits
- A and B are two 4-bit numbers.
- Let A = $A_3A_2A_1A_0$ and Let B = $B_3B_2B_1B_0$
- Inputs have 2^8 (256) possible combinations
- Not easy to design using conventional techniques



MAGNITUDE COMPERATOR

Design of the EQ output (A = B) in 4-bit magnitude comparator

- Define Xi = (Ai Bi)+ (Ai' Bi')
- Thus Xi = 1 IFF Ai = Bi $\forall$ *i =0, 1, 2 and 3*
- Xi = 0 IFF Ai ≠ Bi

Condition for A= B

- EQ=1 (i.e., A=B) IFF

1. A3=B3 → (X3 = 1), and
2. 2. A2=B2 → (X2 = 1), and
3. 3. A1=B1 → (X1 = 1), and
4. 4. A0=B0 → (X0 = 1).

Thus, EQ=1 IFF X3 X2 X1 X0 = 1. In other words, EQ = X3 X2 X1 X0

MAGNITUDE COMPERATOR

- In order to determine whether A>B or B>A, the relative magnitudes of pairs of significant digits starting from the most significant position are inspected.
- If the two digits are equal, the next lower significant pair of digits are compared until a pair of unequal digits is reached.
- If the corresponding digit of A is 1 and that of B is 0, we conclude that A>B.
- If the corresponding digit of A is 0 and that of B is 1, we conclude that A<B.
- To summarize:

(A>B) = A3 B3'+ X3A2 B2' + X3 X2A1 B1' + X3X2X1A0B0'

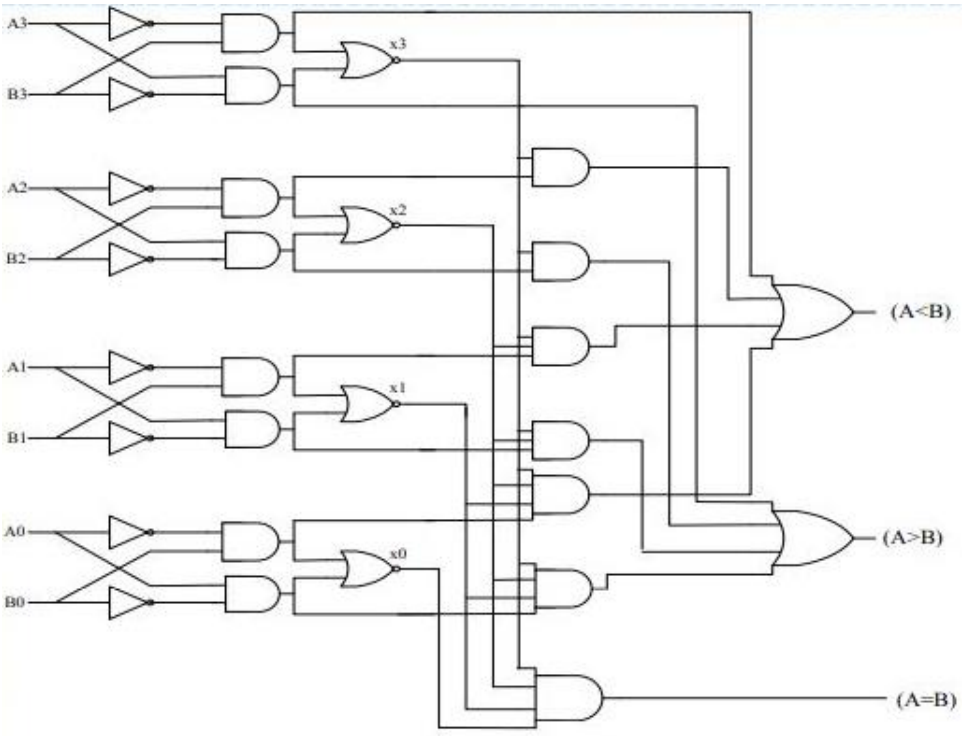
(A<B) = B3 A3' + X3B2 A2' + X3 X2B1 A1' + X3X2X1B0A0'

MAGNITUDE COMPERATOR

Truth table of 4-Bit Comparator						
COMPARING INPUTS				OUTPUT		
A3, B3	A2, B2	A1, B1	A0, B0	A > B	A < B	A = B
A3 > B3	X	X	X	H	L	L
A3 < B3	X	X	X	L	H	L
A3 = B3	A2 >B2	X	X	H	L	L
A3 = B3	A2 < B2	X	X	L	H	L
A3 = B3	A2 = B2	A1 > B1	X	H	L	L
A3 = B3	A2 = B2	A1 < B1	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 > B0	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 < B0	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	H
H = High Voltage Level, L = Low Voltage, Level, X = Don't Care						

MAGNITUDE COMPERATOR

- 4 Bit Magnitude Comparator



4-Bit Magnitude Comparator

DECODERS

- **Definition:**

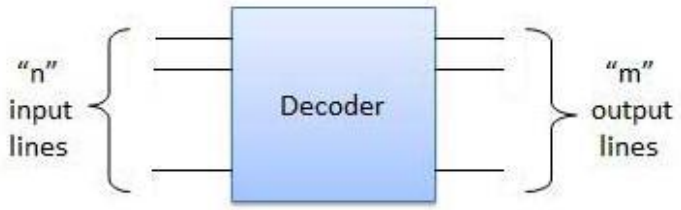
  A decoder is a combinational circuit that converts binary information from **n input lines** to a maximum of **2^n unique output lines**.

- If n-bit decoded information has unused or don't-care combinations, the decoder output will have less than 2^n outputs.

- The decoders presented here are called n-to-m line decoders where $m \leq 2^n$. Their purpose is to generate the 2^n (or less) minterms of n input variables.

DECODERS

- **Application:**
- Decoders are greatly used in applications where the particular output or group of outputs to be activated only on the occurrence of a specific combination of input levels. Some important application of decoder circuit is given below-
- **Address Decoders:** Amongst its many uses, a decoder is widely used to decode the particular memory location in the computer memory system. Decoders accept the address code generated by the CPU which is a combination of address bits for a specific location in the memory.



DECODERS

- **Instruction Decoder:** Another application of the decoder can be found in the control unit of the central processing unit. This decoder is used to decode the program instructions in order to activate the specific control lines such that different operations in the ALU of the CPU are carried out.

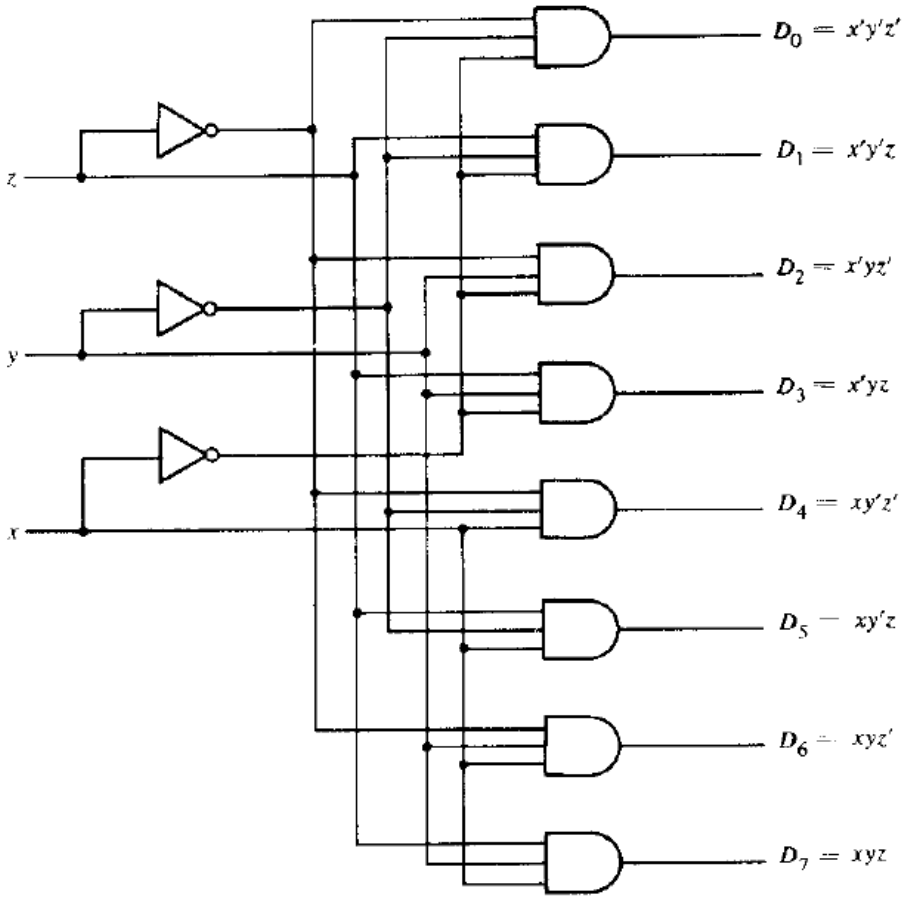
DECODERS

3 X 8 Decoder:

The three inputs are decoded into eight outputs, **each output representing one of the minterms** of the 3-input variables.

A particular application of this decoder would be a **binary-to-octal conversion**. The input variable may represent a binary number and the outputs will then represent the eight digits in the octal number system

DECODERS



*3*8 LINE DECODER*

DECODERS

Truth Table of 3 X 8 line decoder:

From the truth table it is observed that the output variables are mutually exclusive because only one output can be equal to 1 at any one time. The output line whose value is equal to 1 represents the minterm equivalent of the binary number presently available in the input lines.

Inputs			Outputs							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Example 5-2: Design a BCD-to-decimal decoder :

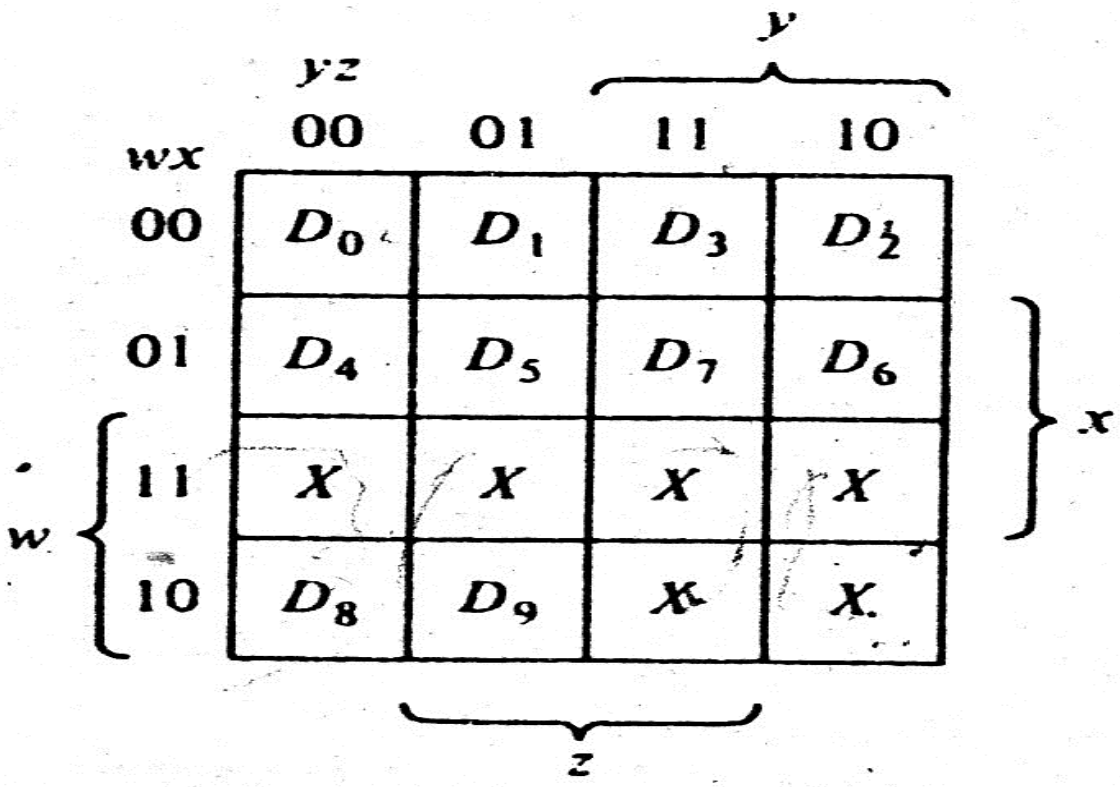
Input BCD				Output Decimal									
w	x	y	z	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1

DECODERS

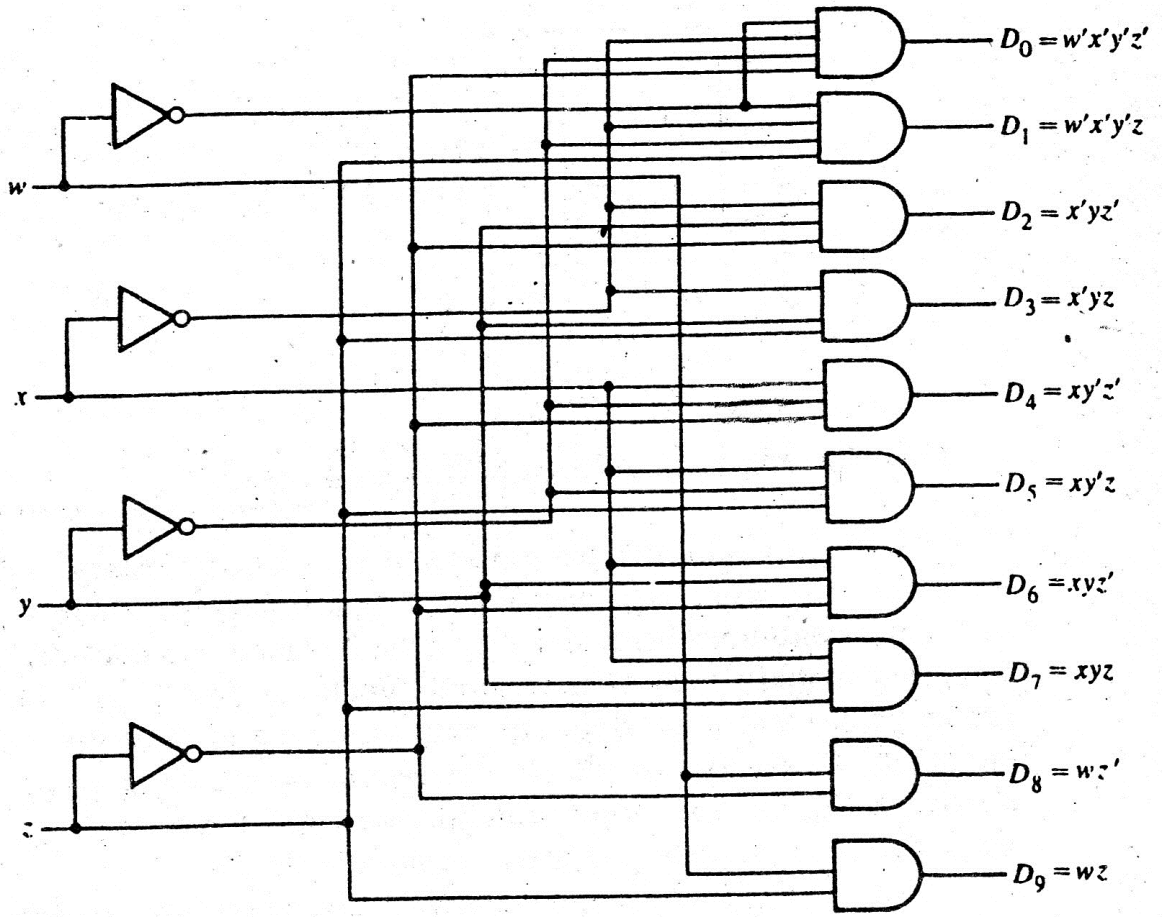
- Since the circuit has ten outputs, it would be necessary to draw ten maps to simplify each one of the output functions, Instead of drawing ten maps, we will draw only one map and write each of the output variables, D0 to d9, inside its corresponding minterm square.
- Six input combinations will never occur, so we mark them as don't care.
- **D0 and D1 is isolated, can't be grouped with don't care. D2 to D7 form pair with their adjacent don't care. D8 and D9 form Quad with their adjacent don't care**
- **Finally we get $D0=w'x'y'z'$ $D1=w'x'y'z$ $D2=x'yz'$ $D3=x'yz$ $D4=xy'z'$ $D5=xy'z$ $D6=xyz'$ $D7=xyz$ $D8=wz'$ $D9=wz$**

DECODERS

- K-map for BCD to Decimal Decoder



DECODERS


w
x
y
z
$D_0 = w'x'y'z'$
$D_1 = w'x'y'z$
$D_2 = x'yz'$
$D_3 = x'yz$
$D_4 = xy'z'$
$D_5 = xy'z$
$D_6 = xyz'$
$D_7 = xyz$
$D_8 = wz'$
$D_9 = wz$

DECODERS

- Combinational Logic Implementation

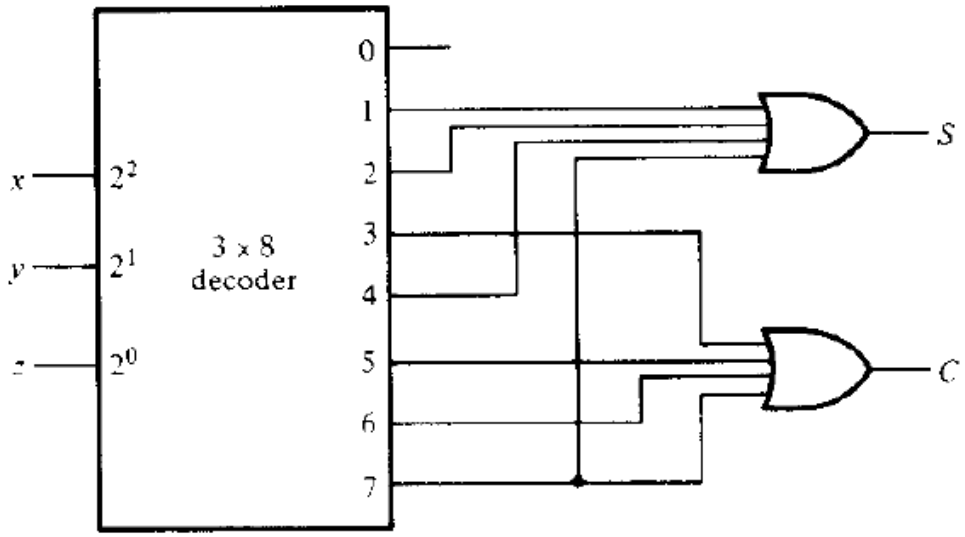
  - Decoder provides 2^n minterms of n input variables.

  - Since any Boolean function can be expressed in sum of minterms canonical form, one can use a decoder to generate the minterms and an external OR gate to form the sum.

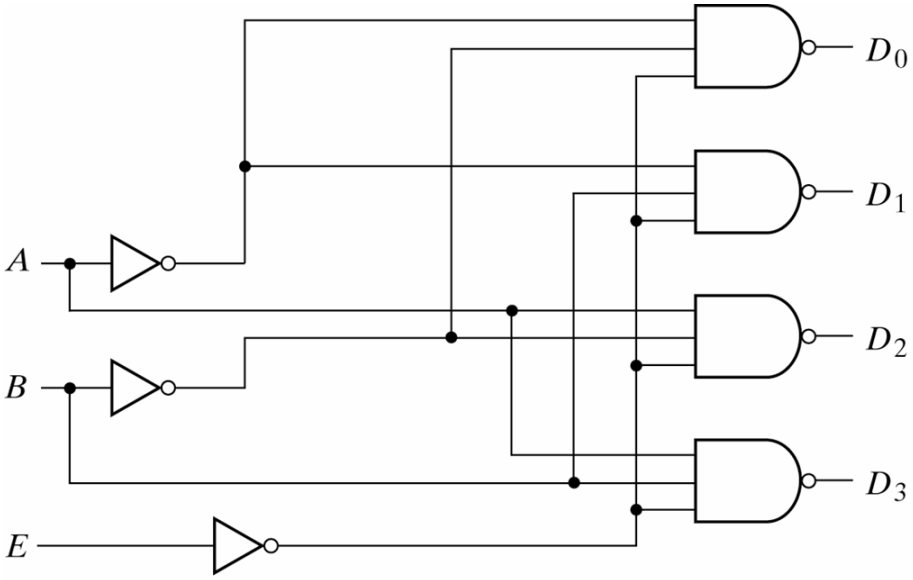
- Example: Implement a full adder circuit with a decoder and two OR gates.

$$S(x, y, z) = \Sigma(1, 2, 4, 7)$$
$$C(x, y, z) = \Sigma(3, 5, 6, 7)$$

DECODERS



DECODER with Enable Input



(a) Logic diagram

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

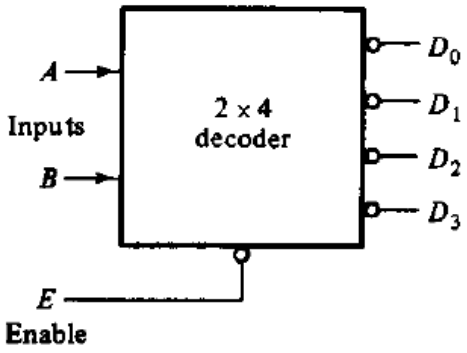
(b) Truth table

DECODER with Enable Input

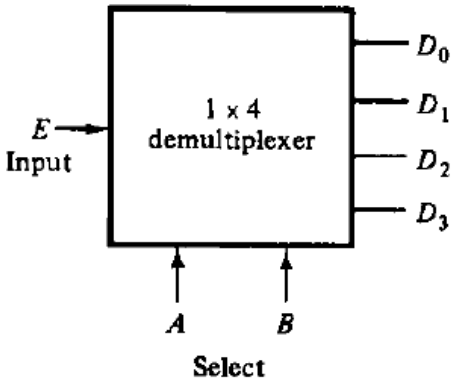
- The circuit operates with complemented outputs and a complement enable input. The decoder is enabled when E is equal to 0.
- Only one output can be equal to 0 at any given time, all other outputs are equal to 1.
- The output whose value is equal to 0 represents the minterm selected by inputs A and B
- The circuit is disabled when E is equal to 1.

DECODER with Enable Input

- A Decoder with Enable input can function as demultiplexer.
- A demultiplexer is a circuit that receives information on a single line and transmits this information on one of 2^n possible output lines.
- The selection of a specific output line is controlled by the bit values of n selection lines.



(a) Decoder with enable



(b) Demultiplexer

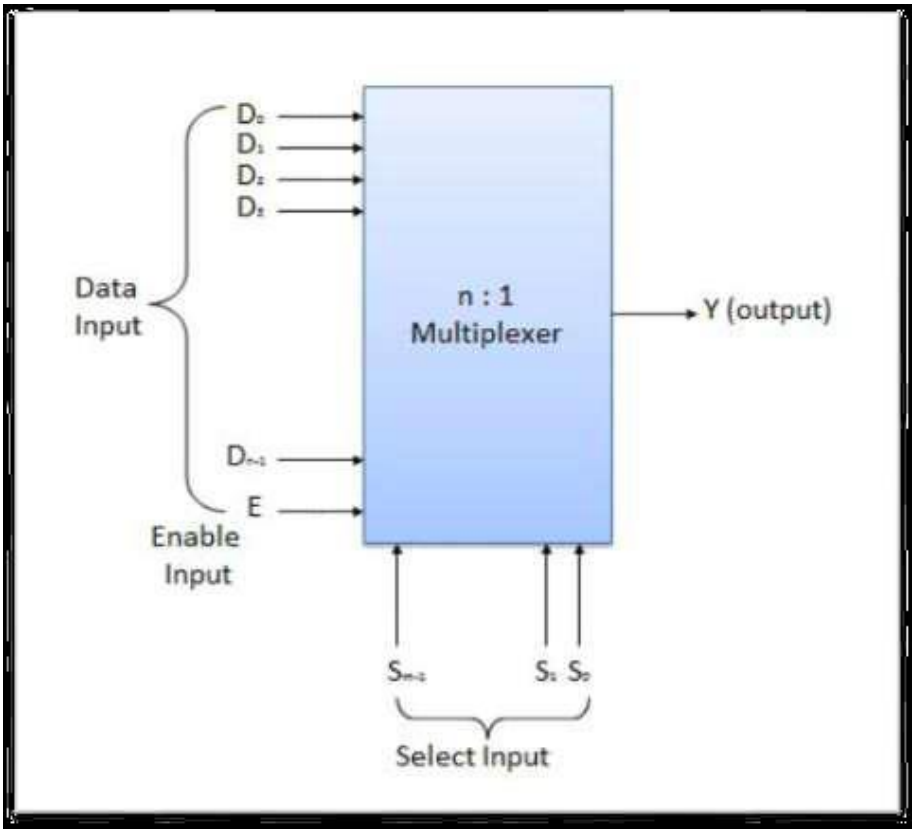
MULTIPLEXER

- A **MULTIPLEXER is a digital circuit that has multiple inputs and a single** output.
- The selection of one of the n inputs is done by the select inputs
- It has one output selected at a time.
- It is also known as **DATA SELECTOR.**
- A multiplexer has
  - N data inputs(multiple)
  - 1 output (single)
  - M select inputs, with $2^M = N$

MULTIPLEXER

- A **MULTIPLEXER is a digital circuit that has multiple inputs and a single** output.
- The selection of one of the n inputs is done by the select inputs
- It has one output selected at a time.
- It is also known as **DATA SELECTOR.**
- A multiplexer has
  - N data inputs(multiple)
  - 1 output (single)
  - M select inputs, with $2^M = N$

MULTIPLEXER

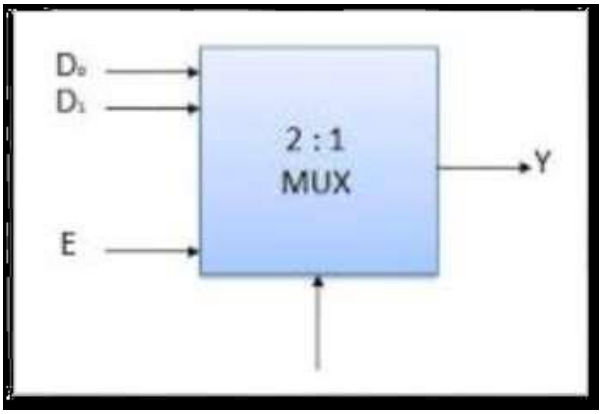


Block Diagram of Multiplexer

MULTIPLEXER

2 to 1 line multiplexer:

Block Diagram



Truth Table

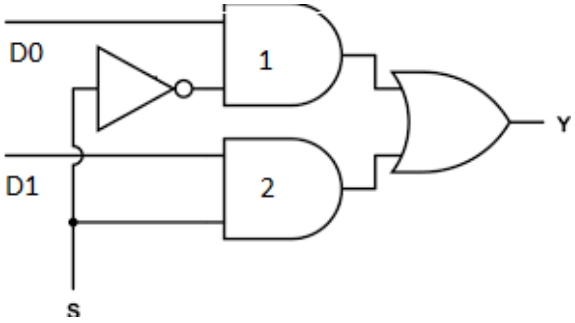
S	OUTPUT Y
0	D0
1	D1

MULTIPLEXER

- The logical level applied to the S input determines which AND gate is enabled, so that its data input passes through the OR gate to the output.
- The output, Y=D0S'+D1S
- When

  - S=0,AND gate 1 is enabled and AND gate 2 is disabled. So, Y=D0

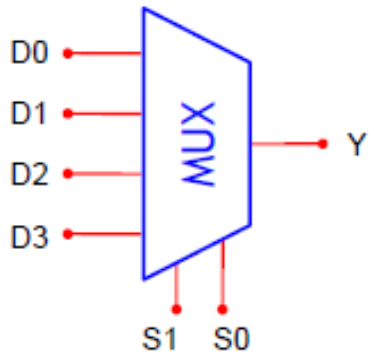
  - S=1,AND gate 1 is disabled and AND gate 2 is enabled . So, Y=D1



MULTIPLEXER

4 to 1 line multiplexer:

Block Diagram

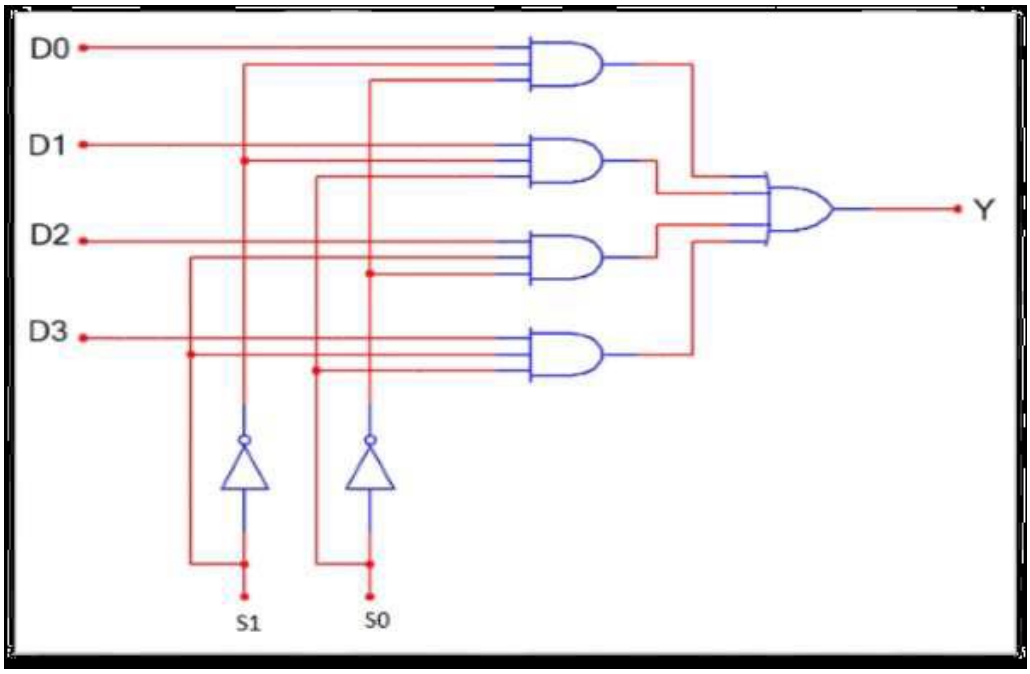


Truth Table

S1	S0	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

MULTIPLEXER

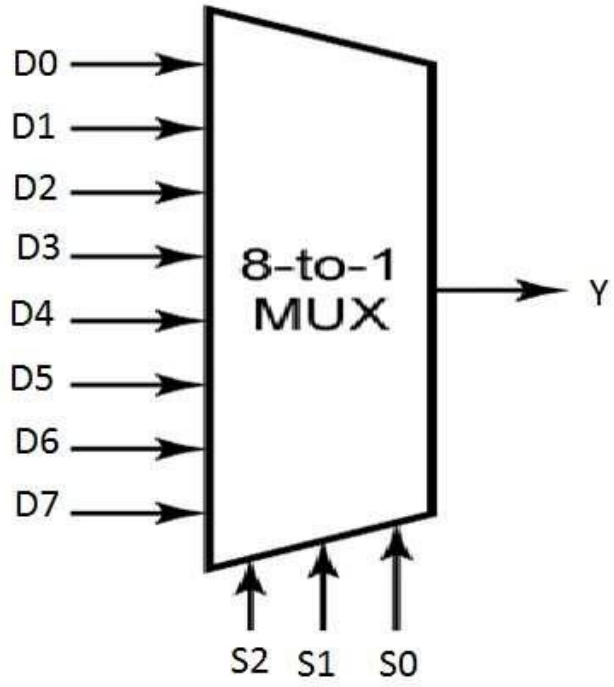
- The logical level applied to the S input determines which AND gate is enabled, so that its data input passes through the OR gate to the output.
- The output, Y=S1'S0'D0+S1'S0D1+S1SO'D2+S1S0D3



MULTIPLEXER

8 to 1 line multiplexer:

Block Diagram



Truth Table

S2	S1	S0	Y
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

MULTIPLEXER

- The logical level applied to the S input determines which AND gate is enabled, so that its data input passes through the OR gate to the output.